103.1043.01

1   This application is submitted in the name of the following inventors:

2   <u>Inventor</u>   <u>Citizenship</u>   <u>Residence City and State</u>

3   Gaurav Banga   India   Sunnyvale, CA

4   Henk J. Bots   United States   Hollister, CA

5   Mark Smith   United States   Cupertino, CA

6

7   The assignee is <u>Network Appliance, Inc.</u>, a corporation having an office at 495

8   East Java Dr., Sunnyvale, CA 94089.

9

10   <u>Title of the Invention</u>

11

12   Prevention and Detection of IP Identification Wraparound Errors

13

14   <u>Background of the Invention</u>

15

16   *1. Field of the Invention*

17

18   This invention relates to reassembly of data fragments of fragmented datagrams in

19   a communication system. In particular, the invention relates to reducing and/or detecting a

20   likelihood of misassembly of data fragments in a communication system utilizing the Internet

21   Protocol (IP) caused by IP identification wraparound.

1   *2. Description of the Related Art*

2

3   The Internet Protocol (IP) has become one of the most widely used

4 communication protocols in the world. IP is part of a layered protocol, which means that another

5 higher level protocol typically uses IP for data communication. Examples of such higher level

6 protocols are the Transfer Control Protocol (TCP) and the User Datagram Protocol (UDP). In

7 addition, even higher level protocols are sometimes utilized, such as the Network File System

8 (NFS). These protocols are well known to those skilled in the art. The protocols are used to

9 send data from a sending station (e.g., a client or a server on a sending end of a communication)

10 to a receiving station (e.g., a client or a server on a receiving end of a communication), possibly

11 through one or more routing devices that form an IP path.

12

13   In order to send a TCP, UDP or other protocol datagram across an IP connection,

14 the datagram is encapsulated in an IP datagram. Often, the IP datagram must be fragmented into

15 plural IP data fragments in order to be sent using the physical network. For example, if a size of

16 the datagram exceeds the physical link's maximum transfer unit (MTU), that datagram must be

17 fragmented into plural IP data fragments with sizes that do not exceed the MTU. Then, a

18 receiving station reassembles the data fragments into the datagram.

19

20   A receiving station determines that data fragments belong to a single IP datagram

21 by looking at an IP identification number in a header of each data fragment. All data fragments

1   from the same IP datagram share the same IP identification number. In addition, the header of

2   each data fragment includes an offset from the start of the datagram, a length of the data

3   fragment, and a flag that indicates whether or not the datagram includes more data fragments.

4   This information is sufficient for reassembly of the IP datagram, which includes the original

5   TCP, UDP or other protocol datagram.

6

7   According to IP, the IP identification number is 16 bits long with a range of 0 to

8   65535. A sending station conventionally uses a simple counter to determine the IP identification

9   number for each IP datagram. In the early days of IP communications, a receiving station most

10   likely would receive all data fragments of a datagram with a particular IP identification number

11   and reassemble the datagram well before this counter could wrap around. If a data fragment was

12   lost, thereby making reassembly of a datagram impossible, all received data fragments of that

13   datagram would be discarded after a timeout of 64 seconds. With the slower communications

14   times that existed in the early days of the IP communications, this timeout was usually sufficient

15   to ensure data fragments would be discarded before the counter at the sending station could wrap

16   around.

17

18   However, today's Internet communications are much faster. Gigabit and 100Mb

19   Ethernet implementations are commonplace, and faster implementations are constantly being

20   developed. As the communications speed increases, the number of IP datagrams sent by a

21   sending station per unit of time also increases. Thus, the simple 16-bit counter conventionally

1   used to generate IP identification numbers wraps around much more quickly. In fact, in a high

2   speed setting, the counter can almost be guaranteed to wrap around within 64 seconds. Thus, a

3   receiving station can receive data fragments from two different IP datagrams that share a

4   common IP identification number before a first one of those datagrams is reassembled.

5

6   Because of the nature of IP communications, it is possible for a data fragment

7   from a second one of two datagrams to arrive at the receiving station before a corresponding data

8   fragment from a first one of the two datagrams. Then, if the two datagrams share a common IP

9   identification number due to wraparound of the sending station's IP identification number

10   counter, the receiving station can misassemble the data fragments. This misassembly can result

11   in corruption of the datagram.

12

13   For example, if first datagram A is fragmented into data fragments A1, A2, A3,

14   A4 and A5, and second datagram B is fragmented into data fragments B1, B2, B3 and B4, it is

15   possible for a receiving station to receive the data fragment B2 before data fragment A2. Then, if

16   datagram A and datagram B share a common IP identification number due to wraparound of the

17   sending station's IP identification number counter, the receiving station can misassemble data

18   fragments A1, B2, A3, A4 and A5 into a datagram, which of course would not contain the proper

19   data.

20

1   Higher level protocols such as TCP and UDP utilize checksums and length checks

2   in an attempt to catch such data corruption. However, the UDP checksum is only 16 bits long. It

3   has been found that in a high speed environment, IP misassembly errors might occur with

4   sufficient frequency that eventually a "false positive" checksum can result. In this case, the

5   checksum can indicate that the UDP datagram has been properly reassembled, while in fact the

6   datagram has been corrupted. Other properties of conventional IP exacerbate this situation, such

7   as IP's acceptance of overlapping data fragments during datagram reassembly. In a UDP

8   communication setting, these types of errors can lead to undetected data corruption. This data

9   corruption might only come to light when the data is actually utilized, a situation that preferably

10   should be avoided.

                                    Summary of the Invention

14   The invention addresses the foregoing concerns by implementing measures

15   designed to reduce a likelihood of misassembly of received data fragments from fragmented IP

16   datagrams. In addition, the invention implements measures designed to detect when a likelihood

17   of such misassembly is high so that appropriate corrective policies can be implemented.

18

19   One embodiment of an aspect of the invention is a method of generating IP

20   identification numbers for IP datagrams. In this embodiment, a plurality of IP identification

21   number generators is maintained. A plurality of receiving stations are associated with the

1   plurality of IP identification number generators such that each receiving station has an IP

2   identification number generator associated therewith. An IP identification number for a

3   datagram sent to one of the receiving stations is generated based on an output of the associated IP

4   identification number generator. This method preferably is performed by an IP layer of a sending

5   station's communication system.

6

7   By using plural number generators, this aspect of the invention slows down

8   wraparound of IP identification numbers used for communication with any given receiving

9   station.

10

11   Preferably, each of the IP identification number generators has at least one

12   receiving station associated therewith. At least one of the IP identification number generators

13   preferably has plural receiving stations associated therewith. In one embodiment, the plurality of

14   IP identification number generators forms an array of number generators such as 16-bit counters.

15   Preferably, the plurality of IP identification number generators is associated with the plurality of

16   receiving stations by hashing destination addresses for the receiving stations and, in one

17   embodiment, protocols for transmitting to those receiving stations so as to form an index to the

18   array. If the hashing includes protocol information, the hashing preferably is performed such that

19   at least half of the number generators in the array are associated with UDP protocol

20   communications.

21

1 An embodiment of another aspect of the invention is a method of reducing a

2 likelihood of misassembly of data fragments from fragmented IP datagrams. In this method, data

3 fragments of a datagram having an IP identification number are received. All received data

4 fragments of the datagram are discarded upon detection of receipt of an overlapping data

5 fragment having the IP identification number, wherein the overlapping data fragment overlaps

6 data in an already-received data fragment. The overlapping data fragment can overlap all or less

7 than all of the already-received data fragment(s). This method preferably is performed by an IP

8 layer of a receiving station's communication system.

9

10 An embodiment of another aspect of the invention also is a method of reducing a

11 likelihood of misassembly of data fragments from fragmented IP datagrams. According to this

12 method, a timeout for reassembling the datagrams is reduced to less than a standard timeout.

13 Preferably, the datagram reassembly timeout is reduced to 45 seconds from the standard timeout

14 of 64 seconds. Alternatively, the datagram reassembly timeout is dynamically reduced based on

15 NFS data for round-trip times between a sending station and a receiving station. This method

16 preferably is performed by an IP layer of a receiving station's communication system.

17

18 Yet another aspect of the invention is embodied in a method of reducing a

19 likelihood of misassembly of data fragments from fragmented IP datagrams. This method

20 includes the steps of receiving data fragments of a datagram having an IP identification number,

21 and reducing a remaining time for reassembling the datagram upon detection of a gap in the

7

1   received data fragments. Preferably, the remaining time for reassembling the datagram is

2   reduced to eight seconds. This method also preferably is performed by an IP layer of a receiving

3   station's communication system.

4   .

5   An additional aspect of the invention is embodied in another method of reducing a

6   likelihood of misassembly of data fragments from fragmented IP datagrams. According to this

7   method, data fragments of a first datagram are received, with the data fragments each having a

8   protocol identification number, a source address, and a first IP identification number. A

9   remaining time for reassembling the datagram is reduced upon detection, before receipt of a last

10   data fragment of the first datagram, of a data fragment of a second datagram having the protocol

11   identification number and the source address but having a second IP identification number.

12   Preferably, the remaining time for reassembling the datagram is reduced to eight seconds. This

13   method also preferably is performed by an IP layer of a receiving station's communication

14   system.

15

16   A further aspect of the invention is embodied in a method of detecting a

17   likelihood of misassembly of data fragments from fragmented IP datagrams. In this embodiment,

18   communication errors between a sending station and a receiving station are detected. The

19   likelihood of misassembly is determined to be high upon detection that the communication errors

20   occur at a high rate for a predefined period of time. The communication errors that are detected

21   can include communication errors detected by an IP layer of the receiving station's

1   communication system. Such IP communication errors include, but are not limited to, receipt of

2   overlapping data fragments and IP datagram reassembly timeout errors. The communication

3   errors that are detected also can include communication errors detected by a UDP layer of the

4   receiving station's communication system. Such UDP communication errors include, but are not

5   limited to, UDP length errors and UDP checksum errors. The communication errors that are

6   detected also can include communication errors detected by an NFS layer of the sending station's

7   communication system.

8

9   Preferably, upon detection that the likelihood of misassembly is high, policies are

10   implemented to reduce the likelihood of misassembly of data fragments. Examples of

11   implementations of such policies include, but are not limited to, preferentially using TCP instead

12   of UDP, using additional checksums and presenting a warning message to a system

13   administrator.

14

15   Another aspect of the invention is embodied in a method for a sending station to

16   detect a likelihood of misassembly at a receiving station of data fragments from fragmented IP

17   datagrams. This method includes the steps of determining a rate at which an IP identification

18   number generator associated with the receiving station wraps around, and determining that the

19   likelihood of misassembly at the receiving station is high upon determination that the IP

20   identification number generator wraps around at faster than a predetermined rate. Preferably, the

1   predetermined rate is once every ninety seconds. Alternatively, NFS re-transmissions also are

2   considered when determining if a likelihood of datagram misassembly is high.

3

4   Policies are preferably implemented to reduce the likelihood of misassembly of

5   data fragments upon determining that the likelihood of misassembly is high. Examples of such

6   policies include, but are not limited to, preferentially using TCP instead of UDP, using additional

7   checksums, and presenting a warning message to a system administrator. When the sending

8   station maintains plural IP identification number generators, such policies also can include

9   reducing a number of receiving stations associated with the IP identification number generator

10   that is wrapping around at faster than the predetermined rate.

11

12   Each of the foregoing methods can be used in conjunction with the others in

13   various combinations to reduce and/or to detect a likelihood of misassembly of IP datagrams.

14   The invention also includes apparatuses such as sending and receiving stations configured to

15   perform the foregoing methods, computer readable code by itself or embodied in a computer

16   program product to cause a computer to perform the foregoing methods, and a memory storing

17   information including instructions executable by a processor to perform the foregoing methods.

18

19   This brief summary has been provided so that the nature of the invention may be

20   understood quickly. A more complete understanding of the invention may be obtained by

1   reference to the following description of the preferred embodiments thereof in connection with

2   the attached drawings.

3

4   Brief Description of the Drawings

5.

6   Figure 1 is a representational view of communication between a sending station

7   and a receiving station across a network such as the Internet.

8   Figure 2 is a representational view of a sending station using plural identification

10   number generators to generate IP identification numbers.

12   Figure 3 is a representational view of a receiving station discarding a datagram

13   upon detection of an overlapping data fragment.

15   Figure 4 is a representational view of a receiving station discarding a datagram

16   upon detection of a partially overlapping data fragment.

17

18   Figure 5 is a representational view of a reduced timeout for reassembling

19   datagrams at a receiving station.

20

1   Figure 6 is a representational view of a receiving station reducing a remaining

2   time for reassembling a datagram upon detection of a gap in received data fragments of the

3   datagram.

4

5   Figure 7 is a representational view of a receiving station reducing a remaining

6   time for reassembling a datagram upon detection of a data fragment from another datagram

7   having that same source address and protocol as the datagram but a different IP identification

8   number.

9

10   Figure 8 is a flowchart for explaining determination that a likelihood of

11   misassembly of datagrams is high upon detection of a high rate of communication errors for a

12   period of time.

13

14   Figure 9 is a flowchart for explaining determination that a likelihood of

15   misassembly of datagrams is high upon determination that an IP identification number generator

16   wraps around at faster than a predetermined rate.

17

18   <u>Description of the Preferred Embodiment</u>

19

20   In the following description, a preferred embodiment of the invention is described

21   with regard to preferred process steps and data structures. However, those skilled in the art

12

1   would recognize, after perusal of this application, that embodiments of the invention may be

2   implemented using one or more general purpose processors or special purpose processors

3   adapted to particular process steps and data structures operating under program control, that such

4   process steps and data structures can be embodied as information stored in or transmitted to and

5   from memories (e.g., fixed memories such as DRAMs, SRAMs, hard disks, caches, etc., and

6   removable memories such as floppy disks, CD-ROMs, data tapes, etc.), with the information

7   including instructions executable by such processors (e.g., object code that is directly executable,

8   source code that is executable after compilation, code that is executable through interpretation,

9   etc.), and that implementation of the preferred process steps and data structures described herein

10   using such equipment and structures would not require undue experimentation or further

11   invention.

12

13   Fig. 1 is a representational view of communication between a sending station and

14   a receiving station across a network such as the Internet. In Fig. 1, sending station 1 sends

15   information across network 2 to receiving station 3.

16

17   Sending station 1 can be a client sending data to a server, a server sending data to

18   a client, or any other device or entity sending data across network 2. Likewise, receiving station

19   3 can be a server receiving data from a client, a client receiving data from a server, or any other

20   device or entity receiving data across network 2.

21

1   A single device, such as a client or a server, can be both a sending station and a

2   receiving station, possibly simultaneously. For example, in typical two-way data

3   communications between a client and a server, the client is a sending station for communications

4   sent to the server and a receiving station for communications received from the server. Likewise,

5   the server is a receiving station for communications received from the client and a sending

6   station for communications sent to the client.

7

8   Sending station 1 communicates through a layered communication protocol.

9   Preferably, the layered communication protocol includes application layer 5, higher level layer 6

10   such as a Network File System (NFS) layer, transport layer 7 such as a Transfer Control Protocol

11   (TCP) layer, User Datagram Protocol (UDP) or other protocol layer, and Internet Protocol (IP)

12   layer 8. Various other combinations of layers are possible. For example, some sending stations

13   do not have higher level layer 6. Also, particular types of layers are designed to work with other

14   types of layers. For example, NFS was originally designed to work with UDP, not TCP. Finally,

15   some applications directly utilize the lower level UDP or IP layers, thereby bypassing much of

16   the error checking (e.g., checksum computations) provided by the higher level and application

17   layers. As data passes through each of the layers from an application program, each layer

18   performs operations on the data such as encapsulation.

19

20   Application layer 5 provides an interface for application programs to send data.

21   Application layer 5 might compute and add a checksum to the data. Such a checksum is useful

14

1   for ensuring data integrity at a receiving station. However, the application layer does not have to

2   use any such checksum.

3

4   Higher level layer 6 such as an NFS layer typically keeps track of network data.

5   This layer also can add a checksum, although such is not mandatory.

6

7   Transport layer 7 packages data in datagrams. Each datagram typically includes a

8   header and data. The data may be of various lengths. The header typically includes source and

9   address information, datagram length, and a checksum. For example, UDP specifies that a UDP

10   datagram has a header with a 16 bit source port number, a 16 bit destination port number, a 16 bit

11   UDP length, and a 16 bit checksum. The checksum is for both the datagram's header and data,

12   as well as for a pseudo-header that includes additional information (IP source address, IP

13   destination address, protocol, and datagram length).

14

15   IP layer 8 encapsulates UDP, TCP or other protocol datagrams into IP datagrams

16   in order to send those datagrams across network 2. Often, an IP datagram must be fragmented

17   into plural IP data fragments in order to be sent across network 2. For example, if a size of the

18   datagram exceeds a known maximum transfer unit (MTU) for network 2, that datagram must be

19   fragmented into plural IP data fragments with sizes that do not exceed the MTU.

20

1   IP layer 8 generates an IP identification number for each IP datagram. All data

2   fragments from the same IP datagram share the same IP identification number. In addition, the

3   header of each data fragment includes an offset from the start of the datagram, a length of the

4   data fragment, and a flag that indicates whether or not the datagram includes more data

5   fragments. This information is sufficient for a receiving station to reassemble the IP datagram,

6   which includes the original TCP, UDP or other protocol datagram. IP datagrams also include a

7   checksum, but only for the header information.

8

9   According to IP, the IP identification number is 16 bits long with a range of 0 to

10   65535. In a high-speed communications setting, a conventional sending station might send many

11   more than 65535 datagrams in a short period of time, causing this IP identification number to

12   wrap around quickly. Thus, a sending station might send data fragments from two different

13   datagrams with the same IP identification number to the same receiving station. This duplicate

14   IP identification number can cause the receiving station to try to misassemble some of these data

15   fragments into a single datagram.

16

17   In a setting where only a UDP data checksum is used to verify data integrity (e.g.,

18   an application checksum is not used or is bypassed and TCP is not used), some of these

19   misassembled datagrams can slip through the weak 16 bit UDP checksum. This problem is

20   exacerbated by the fact that the UDP checksum's strength is data-type dependent, resulting in

21   similar checksums for similar types of data. For example, a corrupt datagram resulting from

1   misassembly of mismatched text data fragments has better than a 1:65535 chance of resulting in

2   a checksum that matches the checksum for the original datagram.

3

4   In order to reduce a likelihood of wraparound of IP identification numbers for

5   datagrams sent to a particular receiving station, an IP layer according to the invention utilizes

6   plural IP identification number generators, as discussed below with respect to Fig. 2.

7

8   In order to help determine when a likelihood of misassembly of datagrams is high,

9   a sending station according to the invention can monitor for a high rate of communication errors

10   that might be the result of datagram misassembly, as discussed below with reference to Fig. 8.

11   The sending station also can monitors the IP identification number generator(s) for rapid

12   wraparound, as discussed below with reference to Fig. 9.

13

14   Returning to Fig. 1, network 2 preferably includes a plurality of routers 10.

15   Examples of network 2 include the Internet, an intranet, an Ethernet network, and any other

16   network or virtual network that utilizes IP communications. The particular configuration of

17   network 2 is representational only of the inclusion of many routers and many possible

18   communication paths through network 2. This configuration has no other significance, and any

19   other configuration that allows communications through network 2 can be utilized with the

20   invention. For example, network 2 could be replaced with a single router 10 between sending

21   station 1 and receiving station 3.

1   Each of routers 10 can have an MTU smaller than the sizes of data fragments sent

2 to that router. If a router receives a data fragment larger than the router's MTU, the router can

.3 further fragment the data fragment. Each data fragment of a datagram can take a different path

4 through network 2. The routers along these different paths can have different MTUs. Thus, data

5 fragments of a single datagram received by a receiving station can have different sizes.

6

7   Receiving station 3 also communicates through a layered communication

8 protocol. Preferably, the layered communication protocol includes layers corresponding to layers

9 in sending stations that might send data to the receiving station. Thus, in Fig. 1, the layered

10 communication protocol of receiving station 3 includes IP layer 12, transport layer 13 such as a

11 TCP, UDP or other protocol layer, higher level layer 14 such as an NFS layer, and application

12 layer 15. Various other combinations of layers are possible, and some applications directly

13 utilize the lower level UDP or IP layers, thereby bypassing much of the error checking (e.g.,

14 checksum verifications) provided by the higher level and application layers. As data passes

15 through each of the layers to an application program, each layer performs operations on the data

16 such as decapsulation.

17

18   IP layer 12 reassembles data fragments into datagrams based on IP identification

19 numbers, length data and flags in the headers of those data fragments. Reassembly time for a

20 datagram is limited by a timeout. If datagram reassembly time from when a first data fragment of

1   a datagram is received exceeds the timeout, all data fragments associated with the datagram are

2   discarded.

3

4   IP layer 12 also verifies a header checksum for received unfragmented datagrams

5   and data fragments, but this checksum only verifies the integrity of the associated IP headers.

6   This checksum therefore does not generally help prevent or detect data fragment misassembly, at

7   least because such misassembly can occur with completely self-consistent IP headers.

8

9   In order to reduce a likelihood of misassembly of data fragments from different

10   datagrams that have the same IP identification number, an IP layer of a receiving station

11   according to the invention can take several actions. The IP layer can discard all data fragments of

12   a datagram if an overlapping data fragment is received, as discussed below with reference to

13   Figs. 3 and 4. The IP layer also can reduce a timeout for datagram reassembly. The overall

14   timeout can be reduced, as discussed below with reference to Fig. 5. In addition, the time for

15   reassembly can be dynamically reduced if a gap is detected in received data fragments of a

16   datagram, as discussed below with respect to Fig. 6, or if a data fragment from another datagram

17   with a different IP identification number is received from the same source with the same

18   protocol, as discussed below with respect to Fig. 7.

19

20   In order to help determine when a likelihood of misassembly of datagrams is high,

21   a receiving station according to the invention can monitor for a high rate of communication

19

1   errors that might be the result of datagram misassembly, as discussed below with reference to

2   Fig. 8.

3

4   Transport layer 8 strips the TCP, UDP or other protocol header off of a datagram,

5   as appropriate. Both TCP and UDP can verify a checksum for the resulting data. However, as

6   noted above, the UDP checksum is relatively weak. It should be noted that the TCP checksum

7   also is not perfect. Corrupt data sometimes passes the TCP checksum, albeit with significantly

8   less frequency than with the UDP checksum.

9

10   The length of the datagram preferably also is verified by the transport layer.

11   However, many length errors are corrected in the IP layer's datagram reassembly. For example, a

12   750 byte data fragment inserted into a space for a 500 byte data fragment during datagram

13   reassembly typically will not result in a UDP or TCP length error because the IP layer truncates

14   overlong data fragments. Thus, length error checking also may not help catch datagram

15   misassembly.

16

17   Higher level layer 14 preferably works in conjunction with higher level layer 6 in

18   sending station 1 to keep track of and to manage network data. Higher level layer 14 also can

19   provide data integrity verification through checksums, although use of such checksums is not

20   mandatory.

21

1   Application layer 15 provides an interface for an application program to receive

2   data. This layer also optionally can provide a checksum and error checking.

3

4   As is evident from the discussion above, many layers of an layered protocol used

5   for network communications can provide checksums and other error detection measures.

6   However, one common method for network communications is to have application programs

7   directly communicate using UDP and IP. The only data checksum in this configuration is the

8   UDP checksum, which is weak enough that it might miss some misassembly of data fragments.

9   The invention provides techniques for decreasing the likelihood of such misassembly, as well as

10 for detecting when a likelihood of misassembly is high.

11

12   Fig. 2 is a representational view of a sending station using plural identification

13 number generators to generate IP identification numbers.

14

15   Briefly, IP identification numbers for IP datagrams are generated. To generate

16 these identification numbers, a plurality of IP identification number generators are maintained. A

17 plurality of receiving stations are associated with the plurality of IP identification number

18 generators such that each receiving station has an IP identification number generator associated

19 therewith. An IP identification number is generated for a datagram sent to one of the receiving

20 stations based on an output of the associated IP identification number generator. Preferably, the

1   IP identification numbers are generated in an IP layer of a sending station's communication

2   system.

3

4   In more detail, Fig. 2 shows sending station 1 with array 17 of N plural IP

5   identification number generators 18. N preferably is a power of two to simplify indexing and

6   hashing, which are discussed below. Examples of N are 16 and 256. Each of IP identification

7   number generators 18 preferably is a 16-bit counter, corresponding to the 16 bits needed for an IP

8   identification number.

9

10   In order to associate a receiving station with an IP identification number

11   generator, sending station 1 preferably uses the receiving station's address. Optionally, sending

12   station 1 also uses the protocol for a particular datagram to be sent to that receiving station.

13   Preferably, the datagram's transport protocol (i.e., TCP, UDP or other protocol) is used for this

14   protocol. As shown in Fig. 2, receiving station address and protocol 20 for a datagram are

15   hashed by hash 21 to form index 22 to array 17.

16

17   In the preferred embodiment, there are more than N possible combinations of

18   receiving station addresses and protocols. In fact, sending station 1 may send data to more than

19   N separate receiving stations. Therefore, more than one receiving station can be associated with

20   each of IP identification number generators 18.

21

1   Hash 21 preferably is designed so that IP identification number generators 18 are

2   distributed fairly evenly among the receiving stations. Preferably, if there are more than N

3   receiving stations, each of plural IP identification number generators 18 has at least one receiving

4   station associated therewith.

5

6   Furthermore, because UDP tends to be more susceptible to datagram misassembly

7   than other transport protocols, hash 21 preferably is designed so that half of IP identification

8   number generators 18 are associated with UDP. The other half of IP identification number

9   generators 18 preferably are associated with all other protocols. Thus, each receiving station

10   preferably will have an IP identification number generator associated therewith for UDP

11   datagrams and an IP identification number generator associated therewith for all other protocol

12   datagrams. This feature of hash 21 can be implemented by including a "UDP/non-UDP" bit in

13   hash 21.

14

15   Whenever sending station 1 needs to send an IP datagram to a receiving station,

16   sending station 1 preferably sends receiving station address and protocol 20 for that datagram

17   through hash 21 to form index 22. Index 22 is then used to index to one of the plural IP

18   identification number generators 18, which provides the identification number and then

19   increments (or vice versa).

20

23

1   By virtue of the foregoing arrangement, a single IP identification number

2   generator is not shared among all receiving stations. Rather, each of plural IP identification

3   number generators is shared only among the associated receiving station/protocol combinations.

4   Wraparound of IP identification numbers for datagrams sent to a particular receiving station

5   using a particular protocol thereby tends to be greatly slowed, reducing a likelihood that data

6   fragments from two datagrams having the same IP identification number will be sent to the same

7   receiving station before reassembly timeout.

8

9   Figs. 3 to 6 are representational views that illustrate various techniques by which a

10   receiving station can further reduce a likelihood of misassembly of data fragments from two

11   different datagrams. In each of these figures, a data fragment is represented by a small box. A

12   letter in the box represents a datagram to which the data fragment belongs, and a number in the

13   box represents the data fragment's position in the datagram. A small numeral to the upper right

14   of each box indicates an order in which the data fragments have been received by the receiving

15   station in each illustrated example. A data fragment that has not been received does not have

16   such a numeral and is designated by a broken outline (see, e.g., data fragment A2 in Fig. 3).

17   Finally, a numeral under each data fragment indicates a size of the data fragment in bytes.

18

19   It should be noted that the details shown in Figs. 3 to 6, such as specific orders,

20   sizes and compositions of data fragments and datagrams, are provided solely to clarify aspects of

21   the invention discussed with respect to each figure and are for illustrative purposes only. The

1 invention is in no way limited to those particular details, as will be apparent to one skilled in the

2 art.

3

4   Fig. 3 is a representational view of a receiving station discarding a datagram upon

5 detection of an overlapping data fragment.

6

7   Briefly, a likelihood of misassembly of data fragments from fragmented IP

8 datagrams is reduced. Data fragments of a datagram having an IP identification number are

9 received. All received data fragments of the datagram are discarded upon detection of receipt of

10 an overlapping data fragment having the IP identification number, wherein the overlapping data

11 fragment overlaps data in an already-received data fragment. Preferably, this technique is

12 performed by an IP layer of a receiving station's communication system.

13

14   In more detail, receiving station 3 in Fig. 3 has received data fragments A1, A3,

15 A4 and A5. Data fragment A2 has not been received. Subsequently, data fragment B1 has been

16 received. Datagrams A and B have identical IP identification numbers in Fig. 3, for example as a

17 result of an IP identification number generator wrapping around in a sending station that sent

18 datagrams A and B. Thus, data fragment B1 overlaps data fragment A1. In other words, if data

19 fragment B1 was assembled into a datagram with data fragment A1, some data from one of the

20 data fragments would overlap data from the other data fragment. This overlapping corresponds

1   to a situation where misassembly can occur, for example if a data fragment B2 was subsequently

2   received. In Fig. 3, data fragment B1 overlaps all of data fragment A1.

3

4   According to the invention, receiving station 3 determines that data fragment B1

5   has overlapped data fragment A1. Upon determining that such overlapping has occurred, a

6   receiving station according to the invention discards all received data fragments for the datagram

7   with the overlapped data fragment. In Fig. 3, the datagram with overlapped data fragment A1 is

8   datagram A, so data fragments A1, A3, A4 and A5 are discarded. The invention similarly would

9   have discard all received data fragments for datagram A if another of its data fragments had been

10   overlapped instead of data fragment A1, for example data fragment A3, A4 or A5.

11

12   By virtue of the foregoing operation, a sending station discards data fragments

13   before misassembly can occur in some situations.

14

15   Fig. 4 is a representational view of a receiving station discarding a datagram upon

16   detection of a partially overlapping data fragment. In Fig. 4, receiving station 3 has received data

17   fragments A1, A3, A4 and A5. Data fragment A2 has not been received. Subsequently, data

18   fragment B2 has been received. Data fragment B2 has a size of 750 bytes, versus the 500 byte

19   size of data fragments A1, A3, A4 and A5. Such a difference in data fragment size can occur, for

20   example, if data fragment B2 traveled across network 2 along a path that had an MTU of 750

1 bytes, while the rest of the data fragments traveled across network 2 along paths with MTUs of

2 500 bytes.

3

4   Datagrams A and B have identical IP identification numbers in Fig. 4. In this

5 situation, the first 500 bytes of data fragment B2 do not overlap any received data fragments of

6 datagram A. However, the last 250 bytes of data fragment B2 do overlap part of data fragment

7 A3. A receiving station according to the invention preferably would detect this overlap and

8 would therefore discard data fragments A1, A3, A4 and A5. Thus, the invention preferably

9 discards data fragments of a datagram when any data in any of those data fragments is overlapped

10 by any data in a subsequently received data fragment with the same IP identification number.

11

12   By virtue of the foregoing operation, a receiving station discards data fragments

13 before misassembly can occur in more situations than if only overlap of entire data fragments

14 was acted upon.

15

16   Fig. 5 is a representational view of a reduced timeout for reassembling datagrams

17 at a receiving station.

18

19   Briefly, a likelihood of misassembly of data fragments from fragmented IP

20 datagrams is reduced by reducing a timeout for reassembling the datagrams to less than a

1   standard timeout. Preferably, this technique is performed by an IP layer of a receiving station's

2   communication system.

3

4   In more detail, receiving station 3 in Fig. 5 has received data fragments A1 and

5   A3 of datagram A and data fragment B2 of datagram B. Data fragment A2 has not been

6   received. Datagrams A and B have identical IP identification numbers. Accordingly, data

7   fragments A1, B2 and A3 could be misassembled into a corrupted datagram as long as the data

8   fragments were all received before a timeout for datagram reassembly using data fragments A1

9   and A3.

10

11   In Fig. 5, time line 23 illustrates a standard IP datagram assembly timeout of 64

12   seconds. Data fragment B2 is received within this time frame, so if this standard timeout was

13   used, misassembly could occur. However, a receiving station according to the invention

14   preferably uses a reduced timeout such as that illustrated by time line 24. This timeout ends

15   before receipt of data fragment B2, preventing any chance of misassembly in the example shown

16   in Fig. 5.

17

18   A timeout of 45 seconds has been found to produce good result in terms of

19   allowing enough time for proper datagram reassembly while preventing some datagram

20   misassembly. Alternatively, the timeout could be determined based on network data for expected

21   communication (e.g., round-trip) times between a particular sending station and a particular

1   receiving station. Such network data preferably could be provided by an NFS layer of each

2   station's communication system.

3

4   Fig. 6 is a representational view of a receiving station reducing a remaining time

5   for reassembling a datagram upon detection of a gap in received data fragments of the datagram.

6

7   Briefly, a likelihood of misassembly of data fragments from fragmented IP

8   datagrams is reduced. Data fragments of a datagram having an IP identification number are

9   received. A remaining time for reassembling the datagram is reduced upon detection of a gap in

10   the received data fragments. Preferably, this technique is performed by an IP layer of a receiving

11   station's communication system.

12

13   In more detail, receiving station 3 in Fig. 6 has received data fragments A1 and

14   A3 of datagram A. Data fragment A2 has not been received, creating a gap in the received data

15   fragments. The gap can be detected by examining the lengths and offsets included in the headers

16   of the received data fragments. This gap indicates that data fragment A2 might have been lost in

17   transit, opening up the opportunity for a data fragment from another datagram to be improperly

18   inserted into this gap during reassembly. Accordingly, a receiving station according to the

19   invention reduces an amount of time left for receipt of the missing data fragment and reassembly

20   of the datagram.

21

1   Reducing the remaining reassembly time to eight seconds in such a situation has

2   been found to produce good results. Eight seconds has been found generally to allow enough

3   time for receipt of a data fragment that has been merely delayed, while generally not allowing

4   enough time for transmission of another datagram with the same IP identification number as the

5   datagram with the gap.

6

7   Of course, if the remaining time before timeout is less than eight seconds, only the

8   remaining time is allowed before timeout. In other words, the remaining time is not increased to

9   eight seconds if it is already less than eight seconds.

10

11   By virtue of the foregoing operation, fewer opportunities for datagram

12   misassembly tend to occur.

13

14   Fig. 7 is a representational view of a receiving station reducing a remaining time

15   for reassembling a datagram upon detection of a data fragment from another datagram having

16   that same source address and protocol as the datagram but a different IP identification number.

17

18   Briefly, a likelihood of misassembly of data fragments from fragmented IP

19   datagrams is reduced. Data fragments of a first datagram are received, with the datagram having

20   a protocol identification number, a source address, and a first IP identification number. A

21   remaining time for reassembling the first datagram is reduced upon detection, before receipt of a

1   last data fragment of the first datagram, of a data fragment of a second datagram having the

2   protocol identification number and the source address but having a second IP identification

3   number. Preferably, this technique is performed by an IP layer of a receiving station's

4   communication system.

5

6   In more detail, one problem with attempting to detect a gap in data fragments is

7   that IP does not provide enough information to directly detect loss of a last data fragment or

8   fragments of a datagram. In particular, IP data fragments indicate if they are or are not a last data

9   fragment. The only indication of relative positions of intermediate IP data fragments are offsets

10   from a start of the datagram. These offsets provide no information about how many data

11   fragments follow a given data fragment. Thus, if a last data fragment of a first datagram is lost, a

12   receiving station only knows that it has received some data fragments of the first datagram but

13   has not yet received a last data fragment. In this situation, the receiving station might receive

14   before timeout a data fragment from another datagram that happens to match the first datagram's

15   IP identification number. This occurrence can lead to misassembly of the first datagram.

16

17   Typically, a sending station will send all of a datagram in a particular protocol to a

18   particular receiving station before sending another datagram in that protocol to that receiving

19   station. Thus, possible loss of a last data fragment of a first datagram can be indicated by receipt

20   of a data fragment from a second datagram sent by the same sending station as the first datagram

21   using the same protocol. A receiving station can tell that the data fragment is from the second

1   datagram by checking for a different IP identification number than that used by the first

2   datagram. Accordingly, in order to help prevent misassembly, a sending station can reduce a

3   time remaining for reassembling a datagram upon receipt of a data fragment from another

4   datagram having that same source address and protocol as the datagram but a different IP

5   identification number.

6

7   Accordingly, receiving station 3 in Fig. 7 has received data fragments A1, A2, A3

8   and A4 of datagram A, but not last data fragment A5. Receiving station 3 has no way of

9   knowing if data fragment A5 is a last data fragment of datagram A. Receiving station 3 has

10   subsequently received data fragment C1 of datagram C. Datagrams A and C have different IP

11   identification numbers. Therefore, no overlapping can occur. Also, data fragments from

12   datagram C will not be misassembled with data fragments from datagram A (barring other

13   processing errors).

14

15   However, because datagram C shares the same source address and protocol as

16   datagram A, it is likely that the last data fragment or fragments of datagram A have already been

17   sent to receiving station 3 and may be lost. Therefore, receiving station 3 according to the

18   invention preferably reduces a remaining time for receipt of the last data fragment or fragments

19   and reassembly of the datagram. Reducing the remaining time to eight seconds has been found

20   to produce good results, allowing for receipt of merely delayed data fragments while still tending

21   to prevent misassembly.

1   The foregoing methods are designed to decrease a likelihood of datagram

2   misassembly. Even if misassembly occurs, UDP and other checksums probably will catch most

3   of the misassembled packets. However, a misassembled packet eventually might slip past the

4   checksums, especially if only UDP checksums are used, with possibly dire consequences for data

5   integrity. Thus, the invention also provides techniques for detecting when a likelihood of

6   datagram misassembly is high so that appropriate corrective action can be taken.

7

8   Fig. 8 is a flowchart for explaining determination that a likelihood of misassembly

9   of datagrams is high upon detection of a high rate of communication errors for a period of time.

10

11   Briefly, a likelihood of misassembly of data fragments from fragmented IP

12   datagrams is detected. In order to detect this likelihood, communication errors between a

13   sending station and a receiving station are detected. The likelihood of misassembly is

14   determined to be high upon detection that the communication errors occur at a high rate for a

15   predefined period of time.

16

17   In more detail, step S801 in Fig. 8 detects if communication errors are occurring

18   at a high rate for a period of time. This error detection can be performed at a sending station or at

19   a receiving station, both with respect to the station itself and with respect to other stations.

20

1   The types of errors indicative of datagram misassembly include IP layer

2   overlapping errors, IP layer timeout, UDP length errors, UDP checksum errors, and NFS errors.

3   Other errors also might be indicative of datagram misassembly.

4

5   IP layer overlapping errors can be flagged by a station's IP layer when overlapping

6   occurs as discussed above with respect to Figs. 3 and 4. Likewise, IP timeout errors can be the

7   result of reduced timeout and reassembly times as discussed above with respect to Figs. 5, 6 and

8   7. UDP length and checksum errors can be the direct result of datagram misassembly that is

9   properly caught by UDP error checking mechanisms. NFS errors can be the result of

10   misassembly errors that slipped through the IP and UDP error checking mechanisms. NFS errors

11   can be implied from an increased rate of NFS re-transmissions.

12

13   Datagram misassembly and situations that create an opportunity for datagram

14   misassembly have been found to create sustained higher rates of one or more of these types of

15   errors. Thus, if such errors are detected, flow proceeds from step S801 to S802, and it is

16   determined that a likelihood of datagram misassembly at the associated receiving stations is high.

17

18   With a high likelihood of datagram misassembly comes an increased chance that a

19   misassembled datagram will pass UDP's weak checksum. Accordingly, flow preferably

20   proceeds from step S802 to step S803, where policies are implemented to decrease the likelihood

21   of datagram misassembly and to increase a likelihood of catching misassembled datagrams.

34

1   Examples of the policies implemented in step S803 include using TCP instead of

2   UDP, if possible. TCP avoids reliance upon IP fragmentation. In addition, additional checksums

3   can be used. These additional checksums can include application checksums that are much

4   stronger than those typically used by communication protocols, possibly incorporating extremely

5   strong hashing functions such as MD5 and SH1. If UDP checksumming is turned off, it can be

6   turned on. NFS, application, and/or other checksums can be utilized, if possible. Furthermore, a

7   warning can be sent to the system administrators of both the sending and the receiving stations so

8   that the source of the errors can be tracked down and corrected.

9

10   Some of these policies may not be possible to implement in every situation. For

11   example, if a server is implementing the policies, the server may not be able to dictate use of

12   additional checksums to a client. Likewise, TCP may not be available between a particular

13   sending station and a particular receiving station. In these situations, policies preferably are not

14   implemented that prevent communications. Of course, if data integrity is essential,

15   communications with a station that is experiencing high error rates can be discontinued.

16

17   Fig. 9 is a flowchart for explaining determination that a likelihood of misassembly

18   of datagrams is high upon determination that an IP identification number generator wraps around

19   at faster than a predetermined rate.

20

1   Briefly, a sending station detects a likelihood of misassembly of data fragments

2   from fragmented IP datagrams sent to a receiving station. In order to detect this likelihood, the

3   sending station determines a rate at which an IP identification number generator associated with

4   the receiving station wraps around. The likelihood of misassembly at the receiving station is

5   determined to be high upon determination that the IP identification number generator wraps

6   around at faster than a predetermined rate.

7

8   In more detail, in step S901 of Fig. 9, a sending station determines a rate at which

9   its IP identification number generator(s) wrap around. In step S902, it is determined if this rate

10   exceeds a predetermined threshold for any particular IP identification number generator. A

11   predetermined threshold of 90 seconds has been found to work well.

12

13   If an IP identification number generator wraps around at faster than the

14   predetermined rate, a possibility exists that two datagrams having the same IP identification

15   number will be sent to a receiving station before the first of the two datagrams times out, thereby

16   creating an opportunity for datagram misassembly. Accordingly, if the threshold is exceeded,

17   flow proceeds to step S903. In step S903, it is determined that a likelihood of datagram

18   misassembly at the associated receiving stations is high.

19

20   Flow then proceeds to step S904, where policies are implemented to address the

21   high likelihood of datagram misassembly. For example, if the sending station is utilizing plural

1   IP identification number generators, the association between the number generators and the

2   receiving stations can be changed so that fewer receiving stations are associated with the number

3   generator that is wrapping around too quickly. With reference to Fig. 2 above, one technique of

4   changing this association is to change hash 21.

5

6   In addition, policies along the lines of those discussed above with respect to step

7   S803 in Fig. 8 also can be implemented. Again, unless data integrity is essential, only those

8   policies that can be implemented without discontinuing communications preferably are

9   implemented.

10

11   *Alternative Embodiments*

12

13   Each of the techniques discussed above can be used in conjunction with the

14   others. For example, a sending station can check for communication errors in conjunction with a

15   high rate of IP identification number generator wraparound. Other combinations of the foregoing

16   techniques are possible. Thus, while preferred embodiments of the invention are disclosed

17   herein, many variations are possible which remain within the content, scope and spirit of the

18   invention, and these variations would become clear to those skilled in the art after perusal of this

19   application.

20   //

21   //